REMARKS/ARGUMENTS

The office action mailed December 23, 2004, has been carefully reviewed and these remarks are responsive to that office action. Reconsideration and allowance of this application are respectfully requested.

Claims 34-48 are pending. Claims 1-33 have been cancelled without prejudice or disclaimer. Claims 39-48 are new.

Claim Rejections - 35 USC § 103

Claims 1-8, 13-19, 21-25, 27-30, and 34-38 were rejected under 35 U.S.C. 103(a) as being unpatentable over Gee et al. (U.S. Patent 6,374,286).

Gee does not establish prima facie obviousness of claim 34 because Gee does not teach or suggest: (1) that the first and second execution threads are application-level-code execution threads that do not execute in a most-privileged CPU mode; and (2) an interrupt service routine that minimizes overhead associated with switching thread execution from the first thread to the second thread.

Claim 34 is directed to a computer-readable medium having computer-executable instructions for performing real-time execution-thread switching comprising: issuing a first non-maskable interrupt from a counter to an interrupt controller when the counter turns over; in response to receiving the first non-maskable interrupt, issuing a second non-maskable interrupt from the interrupt controller to a central processing unit; in an interrupt service routine that services the second non-maskable interrupt, saving a first execution thread's current state information, wherein the first execution thread is an application-level-code execution thread that does not execute in a most-privileged CPU mode, setting the counter to specify when the counter will turn over again, restoring previously stored state information pertaining to a second execution thread, wherein the second execution thread is an application-level-code execution thread that does not execute in a most-privileged CPU mode; and after execution of the interrupt service routine has finished, executing the second execution thread such that the interrupt service routine that services the second non-maskable interrupt minimizes overhead associated with switching thread execution from the first thread to the second thread.

Gee discloses a real-time processor capable of concurrently running multiple independent Java machines. Multiple Java Virtual Machines (JVMs) operate on a single direct execution JAVA processor with each JVM operating in a separate time slice called a partition. Each JVM has its own data and control structures and is assigned a fixed area of memory. Each partition is also allotted a fixed period of time in which to operate, and, at the end of the allotted time, a context switch is forced to another JVM operating in the next partition. The context switch does not transfer control directly from one JVM to another JVM. Instead, at the end of a partition time period control is switched from the currently operating JVM to a "master JVM" during a time period called an "interstice." The master JVM has access to the entire memory space, may execute privileged instructions (such as halt and resume), and handles system interrupts and housekeeping duties. At the end of the interstice time period, the master JVM starts a proxy thread associated with the next JVM to become operational. The proxy thread handles JVM-specific interrupts and checks the status of the associated JVM. If the JVM appears operational, the proxy thread transfers control to the JVM thread. Time intervals such as partition times and interstice times are enforced by hardware timers, and memory accesses are checked by address comparison circuitry to prevent a system failure due to a malfunction in either the master JVM or another JVM. (Abstract; column 23, line 19, through column 26, line 17).

As stated above, at the end of a partition time period, control is switched from the currently operating JVM to a "master JVM," which has access to the entire memory space and may execute privileged instructions (such as halt and resume). If the JVM appears operational, the proxy thread transfers control to the JVM thread. Gee does not, therefore, teach or suggest context switching from a first application-level-code execution thread, which does not execute in a most-privileged CPU mode, directly to a second application-level-code execution thread that does not execute in a most-privileged CPU mode.

As is also stated above, the Master JVM handles system interrupts and housekeeping duties. At the end of the interstice time period, the master JVM starts a proxy thread associated with the next JVM to become operational. The proxy thread handles JVM-specific interrupts and checks the status of the associated JVM. Gee does not, therefore, teach or suggest an

interrupt service routine that minimizes overhead associated with switching thread execution from the first thread to the second thread.

The invention as recited in claim 34 provides a significant functional advantage over the processor disclosed by Gee. By minimizing the overheard associated with switching thread execution, a computer system that executes the operations recited in claim 34 can be more responsive (such as to user input via a keyboard or mouse) relative to a system, such as Gee, that has higher overhead associated with switching thread execution.

For at least the foregoing reasons, Gee does not establish prima facie obviousness of claim 34 because Gee does not teach or suggest: (1) that the first and second execution threads are application-level-code execution threads that do not execute in a most-privileged CPU mode; and (2) an interrupt service routine that minimizes overhead associated with switching thread execution from the first thread to the second thread. Claim 34 is, therefore, in condition for allowance.

Gee does not establish prima facie obviousness of claim 35 because Gee does not teach or suggest that the counter is an advanced programmable interrupt controller. The office action states that column 23, lines 44-55, teaches this limitation. But this cited portion of Gee, which is reproduced below, does not teach or suggest that the counter is an advanced programmable interrupt controller:

> The scheduling of the partition time "slices" is schematically shown in FIG. 12. The JEM processor 1200 interacts with an IOC controller 1210 to manage input and output information as indicated schematically by arrow 1206. Controller 1210 includes a buffer 1212 for communicating with I/O channels 1214. Also included is a DMA unit 1208 which can communicate with a section of a dual port RAM assigned to each of the partitions 1218, 1224, 1230, 1236, 1242 and 1248. I/O controller 1210 handles all interrupts and shuttles I/O data to a dual port RAM section by means of DMA unit 1208. It is also possible that the dual port RAM units could host interpartition mailboxes.

For at least the foregoing reasons, Gee does not establish prima facie obviousness of claim 35 because Gee does not teach or suggest that the counter is an advanced programmable interrupt controller. Claim 35 is, therefore, in condition for allowance.

Gee does not establish prima facie obviousness of claim 38 because Gee does not teach or suggest that the second execution thread is executed after deferred procedure calls, which

were pending when the interrupt service routine finished executing, have been executed. The office action states that column 23, lines 30-33, and column 23, line 67, through column 24, line 13, teach this limitation. But these cited portions of Gee, which are reproduced below, do not teach or suggest that the second execution thread is executed after deferred procedure calls, which were pending when the interrupt service routine finished executing, have been executed:

> After the context switch has occurred, the executive performs housekeeping operations, services any interrupts which have occurred and queues the next partition to assume processor control.
>
> ...
>
> When each partition ends a context switch occurs in order to start the JVM for the next partition. However, the operation of one JVM, for example JVM1 does [sic., not] immediately follow the operation of the preceding JVM. Instead there are time gaps between the partitions. These "gaps" between the partitions are called "interstices" of which interstices 1222, 1228, 1234, 1240 and 1246 are shown. During each interstice time slice, only JVM0 operates and acts to service interrupts and schedule the next JVM for execution. Consequently, two context switches occur at the end of each partition. A first context switch transfers control from the currently operating JVM to JVM0. The second context switch transfers control from JVM0 to the next scheduled JVM.

For at least the foregoing reasons, Gee does not establish prima facie obviousness of claim 38 because Gee does not teach or suggest that the counter is an advanced programmable interrupt controller. Claim 38 is, therefore, in condition for allowance.

Claims 39 and 44 contain limitations that are analogous to the limitations of claim 34 discussed above. Claims 39 and 44 are, therefore, in condition for allowance for at least reasons similar to those discussed above in connection with claim 34.

Claims 35-38, 40-43, and 45-48 are proper dependent claims and are, therefore, in condition for allowance.

CONCLUSION

If any fees are required or if an overpayment is made, the Commissioner is authorized to debit or credit our Deposit Account No. 19-0733, accordingly.

All rejections having been addressed, applicant respectfully submits that this application is in condition for allowance, and respectfully requests issuance of a notice of allowance.

Appln. No.: 09/531,397
Amendment dated April 25, 2005
Reply to Office Action of December 23, 2004

Respectfully submitted,

BANNER & WITCOFF, LTD.

Dated: April 25, 2005

By: Arthur S. Milla 43,905
William J. Klein
Registration No. 43,719

10 S. Wacker Dr., Suite 3000
Chicago, IL 60606
Tel: (312) 463-5000
Fax: (312) 463-5001

WJK/mbo